12. Experimenten met de GPIO-poorten

De uitgangen

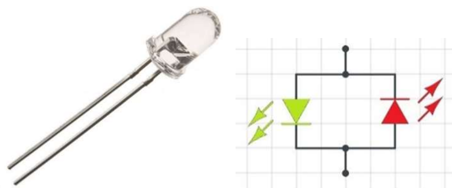
In vorige hoofdstukken hebben wij de GPIO-uitgangen gebruikt voor het laten branden van een LED of het aansturen van een display. De stroom die een poort kan leveren is beperkt, voor de meeste controllers is 12 mA het maximum. Dit is voldoende voor een LED. Hogere stromen kunnen de controller beschadigen. In dit hoofdstuk gaan we poorten voor andere toepassingen gebruiken, toepassingen die soms meer stroom vereisen of die werken met hogere spanningen.

Let op voor kortsluitingen of te hoge stromen aan een uitgangspoort. De controller kan hierdoor beschadigd worden. Ook statische ontlading door het aanraken van de pinnen kan fataal zijn voor de hardware.

Meerkleuren LED's

Het aansluiten en het dimmen van een LED is in een vorig hoofdstuk behandeld. Met een meerkleuren LED zijn meer kleur-effecten mogelijk.

Een **tweekleuren LED** met 2 aansluitingen lijkt uiterlijk op een normale LED in een heldere behuizing. Inwendig bestaat die uit 2 LED's, antiparrallel aangesloten. Een stroom in de ene richting zal één kleur laten oplichten, de omgekeerde stroom toont de andere kleur. *Vergeet de voorschakelweerstand niet bij het aansluiten.* Er zijn verschillende kleurencombinaties in de handel: rood/blauw, rood/groen, …



Figuur 52: tweekleurenled met inwendig schema

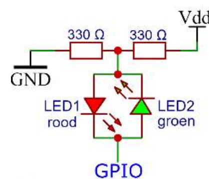
Met wisselstroom kan je de twee kleuren afwisselend tonen. Door de traagheid van ons oog zien we de mengkleur. Op de RPI bereiken we dit effect door de LED met voorschakelweerstand aan te sluiten op 2 poorten. Eén poort 1 en de andere 0 geeft één kleur. De mengkleur krijg je door de poorten snel om te wisselen. Variaties in de mengkleur krijg je door de ene LED langer te laten branden dan de andere. De

kwaliteit van de tussenkleur is sterk afhankelijk van de kwaliteit van de LED. De meeste tweekleuren LED’s zijn niet echt geschikt om tussenkleuren te tonen .

Het programma hieronder laat de LED eerst enkele keren flikkeren tussen de twee kleuren. Daarna zie je de tussenkleur.

```
#-----------------------------#
#  led-bicolor.py             #
#                             #
#  1 MAART 2021               #
#  Dirk Ghysels               #
#-----------------------------#
from machine import Pin
from utime import sleep
led1 = Pin(1, Pin.OUT)
led2 = Pin(2, Pin.OUT)
n = 0
while (n<5):
    led1.value(0)
    led2.value(1)
    sleep(.5)
    led1.value(1)
    led2.value(0)
    sleep(.5)
    n += 1
while (True):
    led1.value(0)
    led2.value(1)
    sleep(.001)
    led1.value(1)
    led2.value(0)
    sleep(.001)
```

Met één poort kan het ook: het schema hieronder toont de aansturing met één poort. De rode LED brandt bij uitgang 0, de groene met uitgang 1.



Figuur 53: 2 kleuren LED aan één poort

Een **tweekleuren LED** met 3 aansluitingen bestaat uit 2 LED’s in één behuizing met een gemeenschappelijke anode of gemeenschappelijke kathode. De middelste aansluiting is de gemeenschappelijke. Verbind de gemeenschappelijke kathode

(common cathod, CC) met GND of de gemeenschappelijke anode (common anod, CA) met Vcc. Zet hier de voorschakelweerstand. De andere met 2 aansluitingen verbind je met een vrije GPIO. Een 1 schakelt de LED aan bij een CC-systeem en schakelt de LED uit bij een CC-systeem. Mengkleuren krijg je door twee poorten te schakelen of te dimmen. Gebruik hiervoor een PWM-signaal, zie paragraaf *Digitale Output*.



Figuur 54: tweekleuren LED, Common Cathod

Een **driekleuren LED** heeft 4 aansluitingen en bestaat uit 3 LED's in één behuizing met een **gemeenschappelijke anode** of **gemeenschappelijke kathode**. Meestal worden hiervoor de kleuren rood, groen en blauw gebruikt, met deze hoofdkleuren kan je elke andere kleur vormen. Deze LED's worden dan ook **RGB-led** genoemd. De langste aansluitdraad is de gemeenschappelijke aansluiting.



Figuur 55: RGB-led, links CA, rechts CC

De aansluiting gebeurt zoals bij de tweekleuren LED hierboven. Met 3 hoofdkleuren heb je uiteraard veel meer tussenkleuren. De driekleurenled van de Pimoroni is verbonden aan poort 18 (rood), 19 (groen) en 20 (blauw). De gemeenschappelijke anode gaat naat Vcc. De LED's gaan aan bij een laag niveau van de poorten. Het programma toon t achtereenvolgens de drie kleuren.

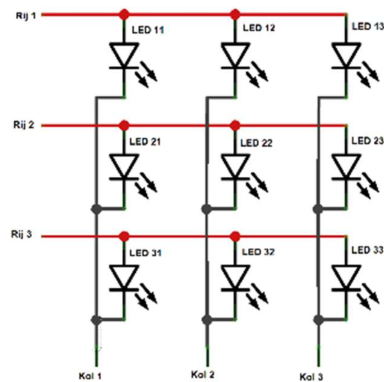
```
#------------------------------#
#  blink-tiny.py               #
#                              #
#  17 maart 2021               #
#  Dirk Ghysels                #
#------------------------------#
from machine import Pin
from utime import sleep
ledR = Pin(18, Pin.OUT)
ledG = Pin(19, Pin.OUT)
ledB = Pin(20, Pin.OUT)

while True:
```

```
        # Rood aan
        ledR.value(0)
        ledG.value(1)
        ledB.value(1)
        sleep(0.3)
        # Groen aan
        ledR.value(1)
        ledG.value(0)
        ledB.value(1)
        sleep(0.3)
        # Blauw aan
        ledR.value(1)
        ledG.value(1)
        ledB.value(0)
        sleep(0.3)
```

Meer LED's

Wil je meer LED's aansluiten, dan kan je ze in een matrix zetten. Deze methode heet **multiplexing**.

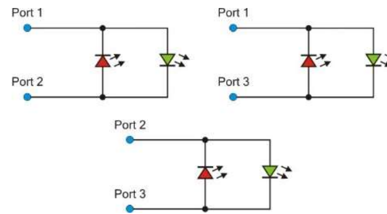


Figuur 56: LED-matrix

LED22 brandt als Rij2 = 1 en Kol2 = 0, de andere rijen maak je 0 en de andere kolommen 1. De rijen en de kolommen verbindt je met GPIO-poorten. De LED-weerstanden kan je aansluiten tussen de rij-aansluitingen en de poorten van de controller. Wil je een LED-patroon tonen, activeer dan eerst de LED's van kolom 1, dan die van kolom 2 enz. Door snel genoeg te wisselen zie je niets flikkeren.

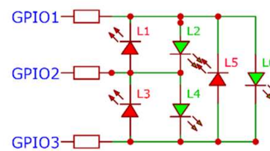
Charlieplexing

Charlieplexing is een methode om zo veel mogelijk LED's aan te sturen met zo weinig mogelijk poorten. Elke LED wordt verbonden met twee poorten. Met drie poorten zijn er zes manieren om een LED aan te sluiten:



Figuur 57: drie poorten - zes LED's

Als we alles aansluiten op GPIO-poorten met de nodige voorschakelweerstanden zien we volgend schema:



Figuur 58: Charlieplexing, 3 poorten

Elke LED kan apart aangesproken worden. Er kunnen geen twee LED's tegelijk branden. De truc bestaat er in de niet-gebruikte poort(en) uit te schakelen door ze te configureren als input. Ze zijn dan hoog-ohmig verbonden met de LED's en hebben geen invloed. LED1 brandt als Port2 = 1, Port1 = 0, Port3 = input. Merk op dat LED4 en LED5 is serie staan tussen Port2 en Port1. Over elke LED staat een spanningsval, 2V voor een rode, 3.4 V voor een groene. De uitgangsspanning van Port2 is echter te klein om LED4 en LED5 te laten oplichten.

LED	Port1	Port2	Port3
L1	0	1	input
L2	1	0	input
L3	Input	0	1
L4	input	1	0
L5	0	input	1
L6	1	Input	0

Deze methode kan veralgemeend worden. Met 4 GPIO-lijnen kan je 12 LED's aansluiten. Met n GPIO's kan je n.(n-1) LED's aansluiten[2]. De methode is hieronder geïmplementeerd in functie charlie. *Het is een omslachtige methode om 6 LED's aan te spreken, maar het kan nuttig zijn als je te weinig GPIO's ter beschikking hebt. (Sommige kleine bordjes zijn schaars bedeeld met GPIO-pinnen)*

```
#-----------------------------#
#  charlieplexing.py          #
```

[2] De Combinatieleer uit wiskunde maakt dit soort berekeningen.

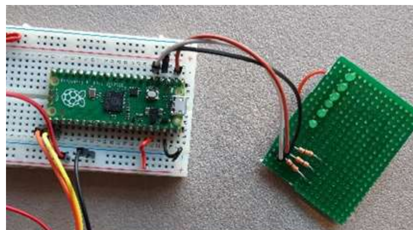
```
#                              #
#  1 MAART 2021                #
#  Dirk Ghysels                #
#------------------------------#
from machine import Pin
from utime import sleep
import utime
led1 = Pin(1, Pin.OUT)
led2 = Pin(2, Pin.OUT)
led3 = Pin(3, Pin.OUT)

def charlie (k):
    if (k==1):
        led1 = Pin(1, Pin.IN)
        led2 = Pin(2, Pin.OUT)
        led3 = Pin(3, Pin.OUT)
        led2.value(1)
        led3.value(0)
    if (k==2):
        led1 = Pin(1, Pin.IN)
        led2 = Pin(2, Pin.OUT)
        led3 = Pin(3, Pin.OUT)
        led2.value(0)
        led3.value(1)
    if (k==3):
        led2 = Pin(2, Pin.IN)
        led3 = Pin(3, Pin.OUT)
        led1 = Pin(1, Pin.OUT)
        led3.value(0)
        led1.value(1)
    if (k==4):
        led2 = Pin(2, Pin.IN)
        led3 = Pin(3, Pin.OUT)
        led1 = Pin(1, Pin.OUT)
        led3.value(1)
        led1.value(0)
    if (k==5):
        led3 = Pin(3, Pin.IN)
        led1 = Pin(1, Pin.OUT)
        led2 = Pin(2, Pin.OUT)
        led1.value(0)
        led2.value(1)
    if (k==6):
        led3 = Pin(3, Pin.IN)
        led1 = Pin(1, Pin.OUT)
        led2 = Pin(2, Pin.OUT)
        led1.value(1)
        led2.value(0)
    return

while(True):
    for k in range (1,7):
        charlie(k)
        print(k)
```

```
        sleep(.5)
```

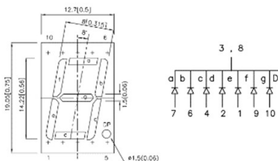
Je kunt de schakeling op een stukje gaatjesprint zetten. Verbind die dan pet GPIO-pinnen 1, 2 en 3. Waarschijnlijk zullen de LED's niet direct in de juiste volgorde oplichten, verwissel de aansluitingen tot alles juist werkt.



Figuur 59: Charlieplexing

7-segment displays

Een 7-segment display vormt een cijfer door een aantal led-staafjes te laten oplichten.



Figuur 60: 7-segment display

De figuur hierboven is overgenomen uit de technische fiche van Kingbright. De kathodes van de staafjes en van het decimale punt zijn verbonden, er bestaan ook modellen met een gemeenschappelijke anode.

Met 4 (of meer) displays vorm je een getal van 4 (of meer) cijfers. Die combinatie stuur je aan met de matrixmethode hierboven. De acht rijen verbindt je met de acht anodes van de modules, rij 1 met alle a-staafjes, rij 8 met alle decimale punten. Het aantal kolommen is gelijk aan het aantal cijfers, de kolommen worden met de gemeenschappelijke kathodes verbonden.

Eer zijn modules te,koop met 2, 3 4 of meer 7-segment cijfers. Een 4 cijfer-module heeft 12 aansluitingen, 7 voor de segmenten, 1 voor de decimale punten en 4 voor de gemeenschappelijke kathodes (of anodes) van de cijfers.



Figuur 61: 7-segment module

Het project hieronder is gemaakt voor een module met 4 cijfers en gemeenschappelijke anode. Het probleem bij deze displays is de lastige bekabeling maar vooral het gebruik van multiplexing. Dat betekent dat het programma voordurend de verschillende cijfers achter elkaar moet tonen zonder stoppen. Het programma toont de getallen van 0 tot 9999, één getal per seconde. Het hoofdprogramma roept functie *toongetal()* op, die houdt zich bezig met het tonen van de cijfers in een oneindige lus. De berekeningen, het tonen van 0 tot 9999 doen we met een timer. Na één seconde zal de interruptfunctie het getal verhogen Variabele *getal* is daarom een globale variabele.

Tooncijfer zet een cijfer *cf* op het plaats *nr*. Parameter dp geeft aan of er al dan niet een decimaal punt moet getoond worden. Variabele c is een list van 8 getallen in binaire vorm. De bits van die getallen geven aan welke segmenten getoond worden. Het eerste getal stelt cijfer nul voor, het laatste de 9. De bits stellen achtereenvolgens dp, segment a, … ,segment g voor. 0 = aan, 1 = uit.

Met een beetje zoekwerk kan je dit programma aanpassen voor modules met gemeenschappelijke kathodes.

Deze methode om cijfers af te beelden is heel omslachtig en de kans op fouten is groot. In een volgend hoofdstuk zien we 7-segment display modules met een seriële aansluiting. Met de bijhorende library is dat een veel eenvoudiger oplossing.

```
#-----------------------------#
#  7-segment.py               #
#                             #
#  2 MAART 2021               #
#  Dirk Ghysels               #
#-----------------------------#
from machine import Pin, Timer
getal = 0
from utime import sleep
import utime

an1 = Pin(1, Pin.OUT)
an2 = Pin(2, Pin.OUT)
an3 = Pin(3, Pin.OUT)
an4 = Pin(4, Pin.OUT)

sega = Pin(6, Pin.OUT)
segb = Pin(7, Pin.OUT)
```

```
segc = Pin(8, Pin.OUT)
segd = Pin(9, Pin.OUT)
sege = Pin(10, Pin.OUT)
segf = Pin(11, Pin.OUT)
segg = Pin(12, Pin.OUT)
segdp = Pin(13, Pin.OUT)

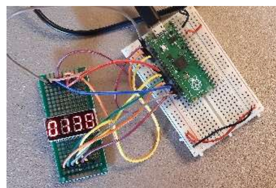

def tooncijfer(cf, nr, dp):
    global an1, an2, an3, an4
    c = [0b00000001, 0b01001111, 0b0010010,
0b00000110]
    c = c + [0b01001100, 0b00100100, 0b00100000,
0b00001111]
    c = c + [0b00000000, 0b00001100 ]
    if (nr==1):
        an1.value(1)
        an2.value(0)
        an3.value(0)
        an4.value(0)
    if (nr==2):
        an1.value(0)
        an2.value(1)
        an3.value(0)
        an4.value(0)
    if (nr==3):
        an1.value(0)
        an2.value(0)
        an3.value(1)
        an4.value(0)
    if (nr==4):
        an1.value(0)
        an2.value(0)
        an3.value(0)
        an4.value(1)

    cs = c[cf]
    segg.value(cs%2)
    cs //= 2
    segf.value(cs%2)
    cs //= 2
    sege.value(cs%2)
    cs //= 2
    segd.value(cs%2)
    cs //= 2
    segc.value(cs%2)
    cs //= 2
    segb.value(cs%2)
    cs //= 2
    sega.value(cs%2)
    cs //= 2
    segdp.value(1-dp)
    return
```

```
def toongetal():
    t = .002
    for k in range (1,10):
        sleep(t)
        c = getal%10
        tooncijfer(c,4,0)
        sleep(t)
        c = (getal//10)%10
        tooncijfer(c,3,0)
        sleep(t)
        c = (getal//100)%10
        tooncijfer(c,2,0)
        sleep(t)
        c = (getal//1000)
        tooncijfer(c,1,0)
        sleep(4*t)
        return

def tik(timer):
    global getal
    getal += 1
    if (getal==9999):
        getal = 0

tim = Timer()
tim.init(freq = 1, mode = Timer.PERIODIC, callback = 
tik)
while(True):
    toongetal()
```



Figuur 62: 7-segment display

Geluid maken met een buzzer

Er bestaan twee soorten buzzers: actieve en passieve. Een actieve buzzer geeft geluid als je er een gelijkspanning op zet. Die gaan we hier niet gebruiken. Een passieve buzzer vereist een specifiek signaal om er tonen uit te krijgen. Een buzzer sluit je aan tussen een GPIO en de massa, de pluskant van de buzzer aan de GPIO, de andere kant aan de massa.



Figuur 63: passieve buzzer

Het volgende programma stuurt een buzzer aan met een PWM-signaal. De pulsen zijn 0,5 ms aan en 0,5 ms uit. (500 microseconden = 0,5 milliseconde). De totale duur van een puls is 1 ms, of 1000 pulsen per seconden. Dat geeft dan de frekwentie van het signaal: 1000 Hz.

```
#-----------------------------#
#  buzzer-PWM.py              #
#                             #
#  20 februari 2021           #
#  Dirk Ghysels               #
#-----------------------------#
from machine import Pin, PWM
from time import sleep
pwm = PWM(Pin(17))
pwm.freq(1000)
pwm.duty_u16(32728)
```

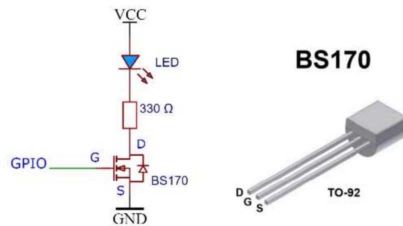
De frekwentie bepaalt de toonhoogte. De dutycycle bepaalt de vorm van het signaal of de klankkleur. Een signaal met een andere dc *klinkt anders*.

De gangbare buzzers werken op 5 volt. De, meeste controllers hebben 3.3 volt poorten. De geluidssterkte is niet indrukwekkend. Je kunt dit verhelpen door de buzzer via een FET op 5 volt te laten werken.

Grotere belastingen schakelen met een FET

De poorten van de gangbare controllers kunnen 3.3 volt leveren met een beperkte stroom. Wil je een belasting aansturen die meer stroom gebruikt of die een hogere voeding vereist, dan kan een FET de oplossing zijn. In de schakeling hieronder gedraagt de BS170 zich als een schakelaar die wordt aangestuurd door de GPIO. Aan de uitgang zie je een LED met voorschakelweerstand, dat kan een 100 mA LED zijn, die heeft een hoger stroomverbruik en hogere lichtopbrengst.

De BS170 is kleine N-kanaal mosfet. Volgens de datasheets kan die 500 mA leveren bij 60 volt. In het schema hieronder schakelt hij aan (hij laat stroom door tussen source en drain) als op de gate en positieve spanning staat.



Figuur 64: schakeling met FET BS170

Je kunt de buzzer uit vorig voorbeeld aansluiten in de plaats van de LED en de weerstand. Verbind de + van de buzzer met een 5V voeding en verbindt de massa van die voeding met die van het bordje.

Een sterkere FET is de IRF520. Die kan 9 ampère leveren en kan tot 100 volt gebruikt worden. Het schema is hetzelfde, alleen adviseert de fabrikant een weerstand van 100Ω aan de ingang (de Gate) toe te voegen. Er bestaan kant en klare modules met de IRF520 of een gelijkwaardige FET die je aansluit op de RPI.



Figuur 65: FET IRF520

Grotere belastingen schakelen met een relais

Met FET's kan je gelijkspanningen schakelen. Wil je ook wisselspanningen schakelen of wil je een galvanische scheiding tussen de RPI en de externe grotere spanningen dan is een relais de oplossing. Op Internet vind je schema's voor de aansluiting van het relais, maar handiger en zeker niet duurder is een volledig opgebouwde module. Hierop zit het relais, de aansturing van het relais met een FET en meestal ook LED's om de status van het relais weer te geven.



Figuur 66: relais-module

Op het relais zie je enkele specifieke kenmerken: het is een 5V-relais, het schakelt met een 5V-spanning. Het relais kan stromen tot 10 ampère schakelen, tot 30 volt gelijkspanning of 250 volt wisselspanning. Let op met die extreme waarden:

- 10 A is veel, de printsporen van de module houden dit waarschijnlijk niet lang vol
- Pas op met netspanning, die is **levensgevaarlijk**

Het aansluiten van de module is simpel: aan de ene kant van de module zie je IN, Vcc en GND. Verbind IN met een GPIO, VCC met 5V van het controller bordje en GND met GND. Een 5 volt relais laat zich aansturen door een digitale uitgang van 3,3 volt. Aan de andere kant zie je een schroefblok met drie aansluitingen, die werken als een wisselschakelaar.

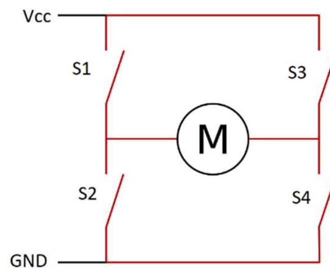
Motoren

Motoren aan-en uitschakelen

Kleine gelijkstroommotoren worden gestuurd met een FET, het schema met de IRF 520 is hiervoor geschikt.

De draairichting aansturen

De polariteit van de aansluitingen van een gelijkstroommotor bepaalt de draairichting. Verander je de polariteit (verwissel de + en de – op de aansluitingen) dan verandert de draairichting. Een H-brug kan de polariteit van de motor omdraaien. Het principeschema zie je hieronder:

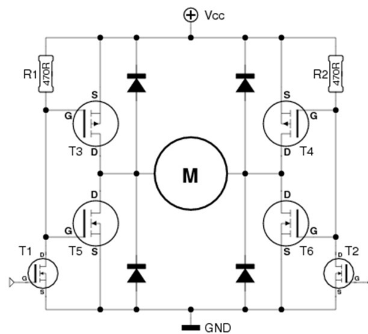


Figuur 67: H-brug, principeschema

Drie of vier gesloten schakelaars geeft kortsluiting. Om de motor te laten draaien moet je twee schakelaars sluiten. Er blijft over:

S1	S2	S3	S4	Motor
1	0	0	1	Draait
0	1	1	0	Draait in andere richting
1	1	0	0	Kortsluiting
0	0	1	1	Kortsluiting
1	0	1	0	Draait niet
0	1	0	1	Draait niet

Er bestaan veel implementaties voor H-bruggen. Je vindt een uitgebreid overzicht op www.prosje.be/schakelen.html . Hieronder zie je een voorbeeld van prosje:

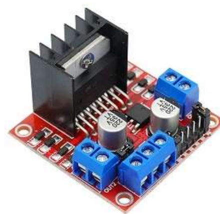


Figuur 68: H-brug met mosfets

T1 en T2 zijn laag-vermogen N-kanaal mosfets, zoals de BS170. De andere vier zijn hoog-vermogen mosfets, T5 en T6 N-kanaal zoals de IRF 520, T3 en T4 P-kanaal zoals de IRF 9520. Een N-kanaal mosfet schakelt bij een positieve spanning op de gate, de P-kanaal bij een negatieve spanning. (negatief t.o.v. de source). De sturing gebeurt met signalen op T1 en T2:

T1	T2	Motor
1	0	Draait
0	1	Draait omgekeerd
0	0	Draait niet
1	1	Draait niet

H-bruggen zijn te koop als module in vele vormen, voor kleine of grote belastingen, voor één of meerder motoren.



Figuur 69: dubbele H-brug voor twee motoren

De snelheid regelen

Op dezelfde website vind je oplossingen om de snelheid van de motor te regelen. Die bestaan er in om de motor te koppelen aan een FET-schakeling en die aan te sturen met een PWM-signaal. Het programma stuurt een PWM-signaal van ongeveer 50% naar de motor.

```
#------------------------------#
#  motor-PWM.py                #
#                              #
#  14 maart 2021               #
#  Dirk Ghysels                #
#------------------------------#
from machine import Pin, PWM
from time import sleep
pwm = PWM(Pin(15))
pwm.freq(100)
pwm.duty_u16(30000)
```

Stuur een motor niet rechtstreeks aan met een controller, die hoge belasting overleeft de controller niet. Zet er een FET-schakeling tussen.

Voor sommige toepassingen draaien de motortjes te snel. Op Internet zijn motortjes te koop een ingebouwde tandwielvertraging. Voor modelbouw auto's bestaan er motortjes met een vertraging en met een wiel.



Figuur 70: modelbouw motortjes

Aanwezigheidsdetectie met een PIR-sensor

Mensen en dieren stralen warmte uit in de vorm van infrarode straling. Een PIR zendt geen straling uit, hij detecteert alleen, vandaar de benaming: PIR = Passieve Infrarood Sensor. Een PIR-sensor detecteert veranderingen van deze straling. Als deze verandering groot genoeg is, geeft de PIR-sensor een puls aan de uitgang.

De sensor-module bestaat uit een IR-sensor, fresnel-lens (de witte kap) die de IR-stralen concentreren naar de IR-sensor en een versterker om het zwakke signaal te versterken. Je sluit de module aan via de drie pinnen onderaan: + wordt verbonden met Vcc van de RP2040, GND met GND en data met een willekeurige GPIO-pin. Onderaan vind je twee potmeters. Met de potmeter links op de foto links hieronder stel je de te tijd tussen detectie van signaal en uitsturen van de puls. Die varieert tussen 2,5 en 250 seconden.De andere potmeter stelt de gevoeligheid in.



Figuur 71: PIR-sensor

PIR-sensoren hebben veel toepassingen: alarmsystemen, automatisch aan- en uitschakelen van verlichting, automatisch openen van deuren, …. Voor een automatische verlichting heb je geen controller nodig. Sluit een voeding van 3 à 5 volt aan op de PIR en verbind de uitgang met een relais die een lamp aanstuurt. Ingewikkelde alarmsystemen werken met intelligentere systemen. Hoe je een PIR

aanstuurt met MicroPython zie je in het voorbeeld hieronder. Bij activatie stuurt het systeem een boodschap door en brandt de LED.

```
#------------------------------#
#  pir.py                      #
#                              #
#  14 maart 2021               #
#  Dirk Ghysels                #
#------------------------------#
from machine import Pin
led = Pin(14, Pin.OUT)
pirPin = Pin(15, Pin.IN)
boodschap = 0
while True:
    if pirPin.value():
        led.value(1)
        if boodschap==0:
            print("alarm geactiveerd")
            boodschap =1
    else:
        led.value(0)
        boodschap = 0
```